平行処理制御方式の可観測性と可制御性(アルゴリズムの数学的基礎理論とその応用)

AUTHOR(S):

上林, 弥彦

CITATION:

上林, 弥彦. 平行処理制御方式の可観測性と可制御性(アルゴリズムの数学的基礎理論とその応用). 数理解析研究所講究録 1986, 591: 95-104

ISSUE DATE:

1986-05

URL:

http://hdl.handle.net/2433/99474

数理解析研究所講究録
第591巻 1986年 95-104

平行処理制御方式の可観測性と可制御性

九州大学工学部　上林弥彦 (Yahiko Kambayashi)

本論文では、平行処理制御方式の可観測性と可制御性という概念について述べ、その応用として、異なる平行処理制御を実現しているようなシステムどうしを統合する問題や平行処理制御における優先順位の導入について考察する。

1. まえがき

大量のデータを扱うには2次記憶は不可欠であり、2次記憶を用いるシステムの効率化として平行処理は特に重要である。とくに、分散システムでは、処理系が複数個存在すること、記憶階層の存在や通信時間の存在により、システム内で同時に複数個の処理を実現しないと、システムの特色を生かせないことから平行処理は不可欠といえる。分散システムの1つの特色として現有のシステムを結合して拡張していける点があり、このためには異種のシステムの統合をどのように扱えばよいかという問題がある。データモデルや言語の異なるシステムの統合についてはすでに多くの研究があるが、平行処理を考えると異なる平行処理制御を採用しているシステムの統合が非常に重要である。まず問題となるのは平行処理制御の効率を落とさないでできるかという点と、もし落ちるとしたら現有の方式をどの程度変更すると良いのか等といった問題がある。この問題は、システム間のプロトコール設計に対しても重要な問題といえる。

2節では、本稿に必要な基本的事項をまとめる。3節では、代表的な平行制御方式である2相施錠方式と時刻印方式について、相互に相手を模擬できるかどうかについて検討する。4節では平行処理制御の可観測性と可制御性という概念を導入して、その実現方法について検討する。5節では、2相施錠方式と時刻印方式の2つの分散システムを統合するための方法について、可観測性と可制御性に基ずく方法について検討している。6節では平行処理制御における優先順位の導入について考察する。

2. 基本的事項

各処理単位(トランザクション)を T_i で表す。データベースに読み書きする基本単位に相当するデータ単位を A, B, C, ... とし、それに対する T_i の書き込み操作を $W_i(A)$, 読み取り操作を $R_i(A)$ で表す。処理単位 T_i は W_i と R_i の系列で表されるものとする。$T_1, T_2, ...$ をある順で逐次的に処理する場合を直列処理と呼ぶ。複数個の処

理単位を少しずつ切り分けて実行することを平行処理と呼ぶが、この場合ある直列処理と常に結果が同じになる場合に直列可能といい、直列可能な場合この平行処理のスケジュールは正しいといわれる。正しいスケジュールを作るために施錠操作が用いられることがあり、この場合はすくみ(デッドロック)が起こりうる。また、直列可能でなくなったりすくみが生じたりした場合、一部の処理単位を無効にして再実行することによってこの問題を扱っている。この無効にする操作を後退復帰といい、ある処理単位の後退復帰によって他の処理単位の後退復帰が生じるときに後退復帰の連鎖と呼ぶ。平行処理制御においては、直列可能性を保障し、すくみの排除、後退復帰の連鎖のできる限りの除去等が重要な問題となっている。

T_iのデータAに対する施錠操作を$L_i(A)$,解錠操作を$U_i(A)$で示す。T_iがデータを読み書きする際にはそのデータが必ずT_iによって施錠されていると仮定すると、W_iとR_iの系列からL_iとU_iの系列を作ることができる。この系列は一意的ではない。しかし、平行処理の立場からはできる限り施錠期間の総和が短くなることが望ましい。あるL_iとU_iの系列が与えられた場合、施錠期間を長くする変換(L_iを前の方へ,U_iを後ろの方へ)を適用することができる。このような変換で前の方にL_iの系列,後ろの方にU_iの系列をまとめた系列が得られる。このような性質を持つ処理単位を2相施錠規約を満足するという。2相施錠規約を満足する処理単位の集合が与えられた場合、施錠による排他制御のみによって直列可能性が保障されることが知られている。しかしながら、すくみを生じるため後退復帰を必要とする。

時刻印方式は、処理単位の到達順に時刻印を付け、この順の直列処理と等価な結果が得られるようにするものである。施錠を用いていないためすくみの問題はなくなるが、直列処理と等価でなくなると後退復帰によって問題を解決するため、書き込みの多い処理の場合には効率が下がると信じられている。各データA_iに対して読み出し時刻印$t_r(A_i)$と書き込み時刻印$t_w(A_i)$とがあり、次の条件を満足しない場合に巻き戻される。ここでT_iの時刻印を$t(T_i)$とする。

(1) データの読み出し　読み出しを行う処理単位をT_iとし、読み出しの対象データをA_jとする。　$t(T_i) > t_w(A_j)$

(2) データの書き込み　書き込みを行う処理単位とT_iとし、書き込みの対象データをA_jとする。　$t(T_i) > t_r(A_j)$
この場合$t(T_i) < t_w(A_j)$なら実際の書き込みはしない。

2相施錠規約、時刻印方式の他に良く知られている平行処理制御の方式として木規約がある。この方式は直列可能性を保障し、すくみも起こらないという特色があり、後退復帰の必要がないが、一般に効率が低いと考えられている。

事後検証法は、読み出しのみからなる処理単位が多い時に有効な方法である。つぎのステップからなる。

(1) 読み取り部分　処理単位が必要とするすべてのデータを作業域にコピーする。

(2) 確認部分　計算が終了した時点で結果の書き込みが直列可能性を満足しているかを確認する。

(3) 書き込み部分　直列可能性を満足していることが分かると作業域にある計算結果を実際のデータの変更に反映させる。

上記の3つの方法は、ある制約のもとで各処理単位が自由に動くものであったが、スケジューラを持ってきて全体として直列可能になるように制御する方法もある。

3. 2相施錠方式と時刻印方式の相互関係

さきに、平行処理制御は効率の向上のために必要であることを述べたが、分散システムでは次に示すように、正しい処理を実現するために不可欠である。このため、異なる平行処理制御方式を用いているシステムを統合した平行処理制御をどのようにするかという問題が生じる

性質：大域的処理が2つしかなく、それぞれが、読み出しだけであり、かつ共有のデータがなくても、局所的処理の存在により、平行処理制御をしなければ、2つの大域的処理の結果が矛盾することも起こりうる。

(証明) 次のような4つの処理単位 T_1, T_2, T_3, T_4 について考える。T_1 と T_2 は大域的処理で読み出し操作のみからなり、T_3 と T_4 は書き込み操作を含む局所的処理である。

T_1：局 S_1 でデータAを読んだ後、局 S_2 でデータDを読む。

T_2：局 S_2 でデータCを読んだ後、局 S_1 でデータBを読む。

T_3：局 S_1 でデータAとBを変更する。

T_4：局 S_2 でデータCとDを変更する。

これらを次のように実行する。

S_1：T_1(Aを読む)　T_3(AとBを変更)　T_2(Bを読む)

S_2：T_2(Cを読む)　T_4(CとDを変更)　T_1(Dを読む)

すなわち、T_1 は $S_1 \rightarrow S_2$ の順で処理され、T_2 は $S_2 \rightarrow S_1$ の順で処理される。この場合、各局における実行順は、データの読み書きの重複によって決まるため、S_1 では $T_1 \rightarrow T_3 \rightarrow T_2$ となり、S_2 では $T_2 \rightarrow T_4 \rightarrow T_1$ となる。この2つの順序は互いに矛盾するため平行処理制御が必要である。

2相施錠方式は施錠を基礎にしており、時刻印方式は施錠機能がない。このような異なる2つの平行制御方式を用いているシステムを統合するための一つの方式として片方の方式を他方が模擬するものが考えられる。

2相施錠方式では、Lの続く施錠段階からUの続く解錠段階に入る時間順が1つの直列スケジュールに対応している。ここでは、後退復帰の連鎖を避けるため全てのデータに対する解錠操作は一括して実行されると仮定する。これを、狭義2相施錠方式という。この方式で時刻印を各処理単位に与え、この時刻印順の直列実行と同じ効果を得るという問題を考える。

(1) 各処理単位に時刻印を与える。

(2) 全データのロック解除を行う前に自分より時刻印の小さな処理単位の実行が全て終了するまで待つ。

(3) すくみが検出された場合、時刻印方式にならい最も小さな時刻印を持つ処理単位を巻き戻す。

上記の方式では、時刻印順のスケジュールと等価にならない場合は全てすくみの形で検出され、2相施錠のすくみ検出機構を用いることができる。

この方法によれば、2相施錠方式のシステムと時刻印方式のシステムとを、全体として時刻印方式で統合することができる。

逆に時刻印方式で2相施錠の模擬を行うのはかなり困難である。これは、施錠という操作を持っていないために、施錠されたデータを用いようとした場合に全て後退復帰によって解決をはかる必要のあるためである。

ある処理単位に非常に大きな時刻印を割り当てると、その処理単位と同じデータを利用する処理はほとんど巻き戻されることになり、この意味で専有利用を実現できる。

もう一つの方法は施錠表を持つもので、各処理単位に施錠・解錠操作に対応する操作を入れ、そこでは施錠表を見て先にすすむかどうか決めるものである。この場合は全ての時刻印を同じにして、時刻印による後退復帰が起こらないようにする。この方式は、共有データとしての施錠表を用いて2相施錠方式を模擬するもので、時刻印方式の特色は全く生かされていない。このため、2相施錠方式で時刻印方式を模擬する方が効率が良いであろう。

この方法で、2つの方式を採用したシステムの結合は次のようになる。

(1) 局所的な処理は、個々のシステムの持つ方式に従う。

(2) 大域的な処理は、時刻印方式で行う。

このように、時刻印方式を1つの標準として、システムを統合する方式が考えられる。

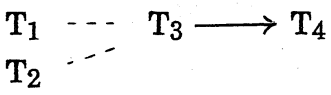
4. 可観測性と可制御性

データベースシステムにおいて、処理単位の順序は、あるデータに対する読み書きの操作の順番によって決められる。

(1) あい続く読み出し操作間では、実行順は決まらない。
(2) あい続く読み出し操作間では、最後のものを除いて実行順は決まらない。

たとえば、あるデータAを、T_1が書き、T_2が読み、T_3が読み、T_4が書き、T_5が読む場合、次のような順序が決まる。

T_1が書き、T_2が書き、T_3が書き、T_4が読む場合は、



ここで、T1とT2の影響は失われてしまうことに、注意を要する。
並行処理制御においては、各データで決まる順序が矛盾しないように制御しなければならない。このため、並行処理制御の可観測性と可制御性という概念を導入する。

可観測性：処理単位の実行順に相当する直列スケジュールないしはスケジュール上の半順序が観測できる。

可制御性：一部の処理単位の間に順序が付いているとき、それに矛盾しないようにスケジュールを決めうる。

スケジューラによるものは可制御性と可観測性を始めから持っている。これらをその機能によって次のように分類する。

[完全可観測]　処理単位の実行した半順序が完全に観測できる場合に完全可観測であるという。

[部分可観測]　処理単位の実行した半順序を満足する半順序ないしは全順序が観測できる場合に部分可観測であるという。

[完全可制御]　任意の半順序にたいして、それを満足するように処理単位の順序が制御できる。

[直列可制御]　任意の全順序にたいして、それを満足するように処理単位の順序が制御できる。

部分可観測性を満足させるには次のようにするとよい。

[2相施錠方式の部分可観測化]　2相施錠方式においては、各処理単位が施錠段階から解錠段階に変わった時刻の順序が1つの直列順となるという性質がある。そのため、必要なデータに対する施錠をすべて終わると直列順をあらわすアレイに処理単位名を書き込むことにより部分可観測化ができる。

[時刻印方式の部分可観測化]　処理終了時の時刻印順が処理単位の満足する全順序となっている。

[事後検証法の部分可観測化]　システムのデータとして保持している。

完全可観測性を満足させるには次のようにするとよい。

[2相施錠方式と時刻印方式の完全可観測化]　各データに対して、半順序を記憶するアレイを用意する。すべてのデータに対する半順序をまとめて1つの半順序をつくる。

事後検証法では、システムのデータとして保持させることが可能で、そうすれば完全可観測にできる。他の方式でも、システム中のログを用いると必要な順序の部分集合を簡単に求めることができるものもある。

システムに手を加える方法は大幅な変更を要する事が多い野で、できれば、手を加えないで部分可観測化をするほうがよい。このとき、次のような性質が利用できる。

(1) 共通のデータを用いないものについては順序は付かない。

(2) 2相施錠の場合は、同時に実行されている処理単位間には順序が付かない。

2相施錠方式の場合、システムに手を加えないで部分可観測化をするには次の方法がある。

[2つの処理単位間の順序の観測]2つの処理単位 T_1 と T_2 の間に共有のデータがあるとし、変数Aの初期値を0にする。T_1 で A=A+1, T_2 で A=2A+2 を実行する。A=0(両方の実行なし), A=1(T_1 のみ実行), A=2(T_2 のみ実行), A=3(T_2, T_1 の順で実行), A=4(T_1, T_2 の順で実行)となるため、Aの値により観測できる。

この方法では、同時に実行される処理単位数の2乗分の変数を用意する必要がある。

次に可制御にするための方法を考える。

[部分的な順序付けのある場合の時刻印方式]T_i と T_j が共通データを利用し、かつ T_i が T_j より先という条件の付いている場合、一番簡単なものは T_i が終わってから T_j を開始するという方式となる。システムの変更が許されると各データに処理単位の利用順を登録しておき、それに矛盾する処理単位の利用を待たせるという方式も可能である。この方法では、使用予定のデータを実際に使わなければすくみが起こってしまう欠点がある。

[部分的な順序付けのある場合の2相施錠方式]T_i と T_j の間で T_i の方が先という制限があるとする。この場合データ a_{ij} を導入し、これに対する施錠によって処理単位の順序を決めるようにできる。T_i と T_j は、次のように変更される。ここで a_{ij} の初期値は0とする。

T_i の変更：a_{ij} に対して専有施錠をする。a_{ij} が0であればこれに1に変える。1であればシステムの故障である。T_i を実行する。a_{ij} の解錠を行う。

T_jの変更： a_{ij}に対して共有施錠をする。a_{ijn}が0であればこれが施錠と解錠を繰り返して1になるまで待つ。1になればT_jを実行する。a_{ij}の解錠を行う。

この方法ではa_{ij}に対して共有施錠より専有施錠を優先するようにシステムを変更しなければならない。上記によりT_iが終わらないとT_jが開始されない。この方法では、推移的な順序に対する制限が不要なので、全順序を保障するには処理単位数だけの変数を用意するとよい。

システムを変更する可制御化は、データの操作履歴をすべて記憶することにより実現できる。この場合矛盾はすべて後退復帰操作によって解決することになり、効率面からは問題が多い。このため、処理単位に次の様な仮定をもうける。

[処理単位に対する仮定] 各データは、たかだた1回読み(R)書き(W)され、両方があるときは、読みの後に書きの順とする。データをそれ以後使わないことが分かるとEとする。

[例] 制御要求が,T_1, T_2, T_3, T_4の順であるとする。あるデータに対する要求は次の通りであると仮定する。

T_1;R　　T_2;W　　T_3;R　　T_4;R

T_2のWは、T_1がRでも実行できる。T_4のRは、実行できずT_3がWかEになるまで待たなければならない。

5. システムの可制御性や可観測性を考えた平行処理制御の統合

3節で考えた方法は、基本的に一つの平行処理制御方式を変更するという立場からのものであったが、ここではより一般的に、平行処理制御方式に標準的な機能を持たせて、これによって統合する方式を考える。

スケジューラによるもの,時刻印を用いるもの,2相施錠を用いるもののうち、2種の方式を混ぜて用いる分散データベースシステムにおける質問の処理は、2相施錠→時刻印, 時刻印→2相施錠, スケジューラ→時刻印 等という形で処理されるのが望ましく、この両システムの間を何度も行き来するというのは処理が複雑となる可能性が高く望ましくない。この場合、両者の間での処理単位の処理順に矛盾がおこらないようにしなければならない。

可制御性と可観測性の組合わせとしては、次の3種が考える。

① 可制御 － 可制御
② 可観測 － 可制御
③ 可観測 － 可観測

①の場合は、中央システムがスケジュールを決め、両方のサブシステムのスケジュールを同一にすることにより全体の直列可能性を保障できる。

②の場合は、片方のシステムで実行しそれを観測して、その情報を他方のシステムの制御に用いることにより全体の直列可能性を保障できる。

③は両方のシステムで実行させ、互いに矛盾が生じると後退復帰を行い、矛盾がなくなるまで切り返すものである。

③の方式は実用上問題が多い。可観測にするよりも可制御にするほうがオーバヘッドが大きいので、これらの中では②の方式の効率が最も高いと思われる。

次にシステムの統合方式について検討する。スケジューラを用いているシステムとの統合は容易であるので、2相施錠方式と時刻印方式との統合について考える。可観測性 – 可制御性の組み合わせで(a)や(b)のような方法がある。

(a) 2相施錠を採用している分散システムで処理をした後時刻印方式のシステムで処理

(1) 2相施錠システムを可観測とし時刻印システムを可制御性として結合する。しかし、時刻印システムを可制御にすると時刻印方式の特色が失われると考えられる。

(2) 2相施錠システムで観測した順序で時刻印を生成し時刻印方式システムで用いる。この場合後者での後退復帰が前者の後退復帰を引き起こすという問題がある。

(3) 2相施錠システムで全必要データを施錠した後時刻印システムへ送る。時刻印システムで実行後2相施錠へ戻って施錠解除する。2相施錠で同時に全データの施錠を終了している処理単位どうしは順序に関する制約がないので、時刻印の部分を全く制御しなくてよいのがこの方式の特色で、時刻印方式システムでの後退復帰により影響も2相施錠システムの実行が終了していないので問題はない。

(3)の方法が最も実用的と考えられる。

(b) 時刻印方式のシステムで処理した後2相施錠のシステムで処理

(1) 全順序を用いる方法　時刻印方式システムでは、終了時の時刻印順が全順序となるので、これを用いて2相施錠システムの制御を行う。前述したように、処理単位数だけの変数を用意するとよい。

(2) 半順序を用いる方法　全順序より半順序を使う方が効率を上げることができるが、時刻印システムの変更を必要とする。

6。処理単位の優先順位

可制御性は、処理の優先順位を与える上で重要といえる。優先順位は、次の方法で実現できる。

A. 後退復帰に際して優先順位の低いものを選ぶ

B. システムで実行する順序を、できる限り優先順位に従うようにする。
C. 可制御性を利用する。
可制御性の利用は、システムの変更を伴うので、本節では、AとBの方法について簡単な例を示す。
[例] 優先順位が次の形式で与えられたとする。

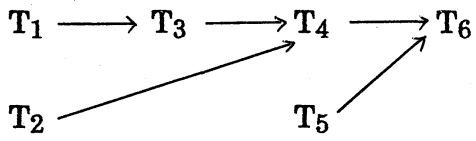
優先順位1のもの　　T_1 T_3
優先順位2のもの　　T_2 T_4
優先順位3のもの　　T_5

この場合、後退復帰に際して優先順位の低いものを選ぶようにするとよい。2相施錠方式　　ですくみが生じた場合、すくみが$T_1T_3T_5$の間で生じているときは、T_5を後退復帰させる。時刻印方式では、一般に時刻印の小さい処理単位が後退復帰させられるが、これをデータの操作履歴を用いて、任意の処理単位を後退復帰できるようにしなければならない。

次の例は優先順位の木とデータの競合関係を考えて、システムに投入する順序を決めるものである。
[例] 優先順位に幅を持たせたものも考えられる。つぎの図の場合、優先順位は次のようになる。

T_1;1　　T_2;1-2　　T_3;2　　T_4;3　　T_5;1-3　　T_6;4

(1) すぐに実行できる最大数の処理単位を実行する。
(2)上記を除いて同じ操作を繰り返す。



実際の場合優先順位の高いものは少ないので、その処理単位のみを対象とした制御を行う方法が考えられる。

謝辞　　本研究は著者が京都大学在職中に始めたものであり、御指導御検討いただいた矢島脩三教授ならびに修士論文としてこの問題を扱っていただいた近藤誠一氏(現在 三菱電機)に深謝します。なお本研究は文部省科学研究費特定研究によるものである。

文献

1) Bernstein, P. A. and Goodman, N., "Concurrency Control in Distributed Database Systems," ACM Computing Surveys, 13 June 1981
2) Eswaran, K. P., Gray, J. N., Lorie, R. A., and Traiger, I. L., "The Notions of Consistency and Predicate Locks in a Database System," CACM 19, 11 Nov. 1976
3) 近藤, 上林, 矢島, "分散データベースにおける異なる平行処理制御方式の統合," 第27回情報処理学会大会, 6K－2, 昭和58年後期
4) Kambayashi, Y. and Kondoh, S., "Global Concurrency Control Mechanisms for a Local Network Consisting of Systems without Concurrency Control Capability," Proc. National Computer Conference, 1984
5) 上林, 近藤, "平行処理制御方式の統合," 第31回情報処理学会大会, 2B－4, 昭和60年後期
6) Papadimitriou, C. H., "The Serializability of Concurrent Database Updates," JACM 26, 4, Oct. 1979